Datos del Proyecto Final de Carrera	
Autor:	Guillermo Solano Méndez
Tutor:	Mariet Theune
Institución donde se realizó:	Universidad de Twente (Holanda)
Coordinador Académico:	José María Sierra
Cotutor:	David Griol
Fecha de Lectura:	18 de Marzo del 2011
Tribunal:	Mariet Theune, Dennis Reidma, Rieks op den Akker
Calificación obtenida:	9 – Sobresaliente

1. Introducción de la investigación

Este es una breve descripción en castellano del proyecto de fin de carrera realizado por Guillermo Solano Méndez durante su estancia como estudiante de intercambio en una universidad de Twente, en Holanda. El título de dicho proyecto fue “conectando el agente conversacional Elckerlyc con el programa Psychometer”. Debido a que este resumen tiene un carácter reducido, básicamente se incluye una pequeña descripción del proyecto y el diseño, los resultados del proceso de validación y las conclusiones finales. Para profundizar se recomienda encarecidamente la lectura del documento de investigación original en inglés.

1.1 Introducción

El objetivo de esta investigación es añadir un avatar virtual con rostro y un conjunto de expresiones y emociones al programa Psychometer en su segunda versión. El Psychometer es un programa basado en turnos que implementa la teoría de la personalidad de los cinco factores para medir la personalidad del usuario. El Psychometer, en su versión primera, fue desarrollado por Marcel Bodewitz como parte de su tesina de Máster en el año 2004. La segunda versión, realizada por Pascal Touset, permite al usuario responder con un cierto lenguaje natural acotado (en inglés u holandés), anteriormente empleando un sistema de marcado de respuesta entre cinco opciones posibles.

El Psychometer utiliza la salida estándar (pantalla) para imprimir las preguntas realizadas al usuario y que le permite clasificar el rasgo del usuario y, tras un set de preguntas-respuestas, definir la personalidad del usuario mediante la teoría de los cinco factores. Las respuestas del usuario se responden con teclado, saltando a la siguiente pregunta tras insertar una pregunta aceptada como válida. Ello implica una interacción basada en turnos hasta que el Psychometer tiene suficientes respuestas para clasificar al usuario, obtenidas a lo largo de varias sesiones (una sesión dura diez minutos).

En el desarrollo de este proyecto la entrada no cambia y se siguen introduciendo las respuestas por teclado, pero la salida emplea un Agente Conversacional Embebido (conocido por sus siglas en inglés ECA). El ECA empleado en esta práctica se conoce cómo Elckerlyc, un proyecto de software libre en fase de desarrollo en la universidad de Twente en Holanda. Elckerlyc acepta muchas posibles configuraciones y emplea un lenguaje de comportamientos, también en desarrollo desde el 2006, que se conoce como Lenguaje de etiquetas de comportamientos ("Behaviour Mark-up Language" o BML). Este proyecto fue la primera vez en la que Elckerlyc se usó como salida de otro programa, recibiendo diferentes entradas de información para generar posibles expresiones en función de cada situación específica. Elckerlyc es un sistema muy potente y configurable, pero dado su estado de desarrollo, carecía de algunas áreas críticas como documentación, control sobre las expresiones faciales, existencia de puntos espaciales para orientar la mirada, errores, implementaciones pendientes, etc. Como consecuencia de ello, este proyecto no sólo consistió en conectar ambos programas y desarrollar las expresiones del avatar proporcionado por Elckerlyc. Si no que además implicó una inmersión en el Proyecto Elckerlyc para modificar código, depurar varías secciones, localizar errores y adaptar el entorno de Elckerlyc. Todas estas modificaciones, adaptaciones y problemas surgidos se encuentran explicados en detalle en otras secciones en esta tesina.

Este proyecto iba a estar, inicialmente, compuesto de dos fases diferenciadas. La primera fase consistía con el proceso de desarrollo de los comportamientos relacionados con la mirada y el habla (salida del Psychometer). La segunda fase, no implementada, pretendía extraer características del set de preguntas (y relaciones entre ellas) para luego decidir posibles expresiones emocionales podrían ser identificadas con el tipo de pregunta que está siendo realizada en ese momento (mediante la búsqueda de palabras y sinónimos que tasaran con sentimientos con el fin de lograr una representación facial acorde a ese sentimiento). Desafortunadamente, debido a limitaciones temporales durante el desarrollo del proyecto y el tiempo que se debió invertir en el proyecto Elckerlyc este proyecto finalizó únicamente con la fase uno; es decir, un set de expresiones y un habla que acompaña al avatar ofrecido por Elckerlyc en un entorno virtual (un despacho de psicología). Con el CD que contiene esta Tesina se incluye un video de demostración del sistema en funcionamiento.

Un objetivo secundario durante la implementación ha sido lograr que la interacción de la ECA fuera independiente de la entrada ofrecida lo máximo posible. Esto ha sido logrado mediante una implementación modulada que captura y controla el flujo de ejecución por lo que se ha evitado cambiar el código del Psychometer lo máximo posible.

Lo a lo largo de la tesina en el proyecto es descrito en diversas secciones. La segunda sección explica los algoritmos implementados para los comportamientos que controlan la mirada, los diferentes comportamientos implementados, limitaciones impuestos en el proyecto y otras variables a tener en consideración para futuras adaptaciones o proyectos similares. La sección tres muestra la parte del diseño del proyecto, con los diagramas de descomposición y aclaraciones del código implementado. La sección cuarta explica en detalle el desarrollo del entorno Elckerlyc final utilizado (adaptaciones, modificaciones, ampliaciones). La quinta sección explica el desarrollo del proyecto así como los principales problemas encontrados (en especial respecto al entorno Elckerlyc debido a que todavía se encuentra en fase de desarrollo). La penúltima sección, la sexta, se centra en el proceso de validación realizada y los resultados obtenidos. Por último, la última sección explica las conclusiones del proyecto.

En pocas palabras los objetivos de la investigación eran simular el comportamiento humano que se resume en “los hablantes tienen tendencia de aparta la mirada de quien escucha en puntos críticos de cambio de turno. Quien escucha suele mostrar atención continua mientras el hablante está en proceso de conversación, decayendo esta atención en función de la longitud de la conversación sin cambiar de tema o del interés de quién escucha”. Esta idea simplificada es la funcionalidad que se ha buscado en el algoritmo implementado principal.

1.2 Diseño del sistema

Este apartado es un breve resumen del diseño del sistema. Para ver el diseño en profundidad leer la sección “3 Design of the Project, modules and implementations” en la tesina.

El Psychometer original fue diseñado empleando una arquitectura Modelo-Vista-controlador, independizando la lógica del usuario de la interfaz. Se ha aprovechado la comunicación entre el controlador y la vista, mediante una clase que actúa de puente, para obtener la salida deseada que será procesada para generar los comportamientos deseados posteriormente en Elckerlyc. El diseño final obtenido hace independiente un programa del otro, de manera que si el Psychometer fuera sustituido por otro programa que ofreciera unas salidas similares (preguntas y basado en turnos) dicho puente de comunicación seguiría funcionando. De hecho, al emplear Elckerlyc el estándar BML se podría añadir una ECA diferente que procese este lenguaje mediante la simple modificación de la clase que comunica el proyecto con el realizador de Elckerlyc (componente Elckerlyc Connection en el diagrama).

A continuación se muestra el nivel cero del diagrama de descomposición y una breve descripción de cada módulo. Para una descripción más profunda se debe leer la tesina.

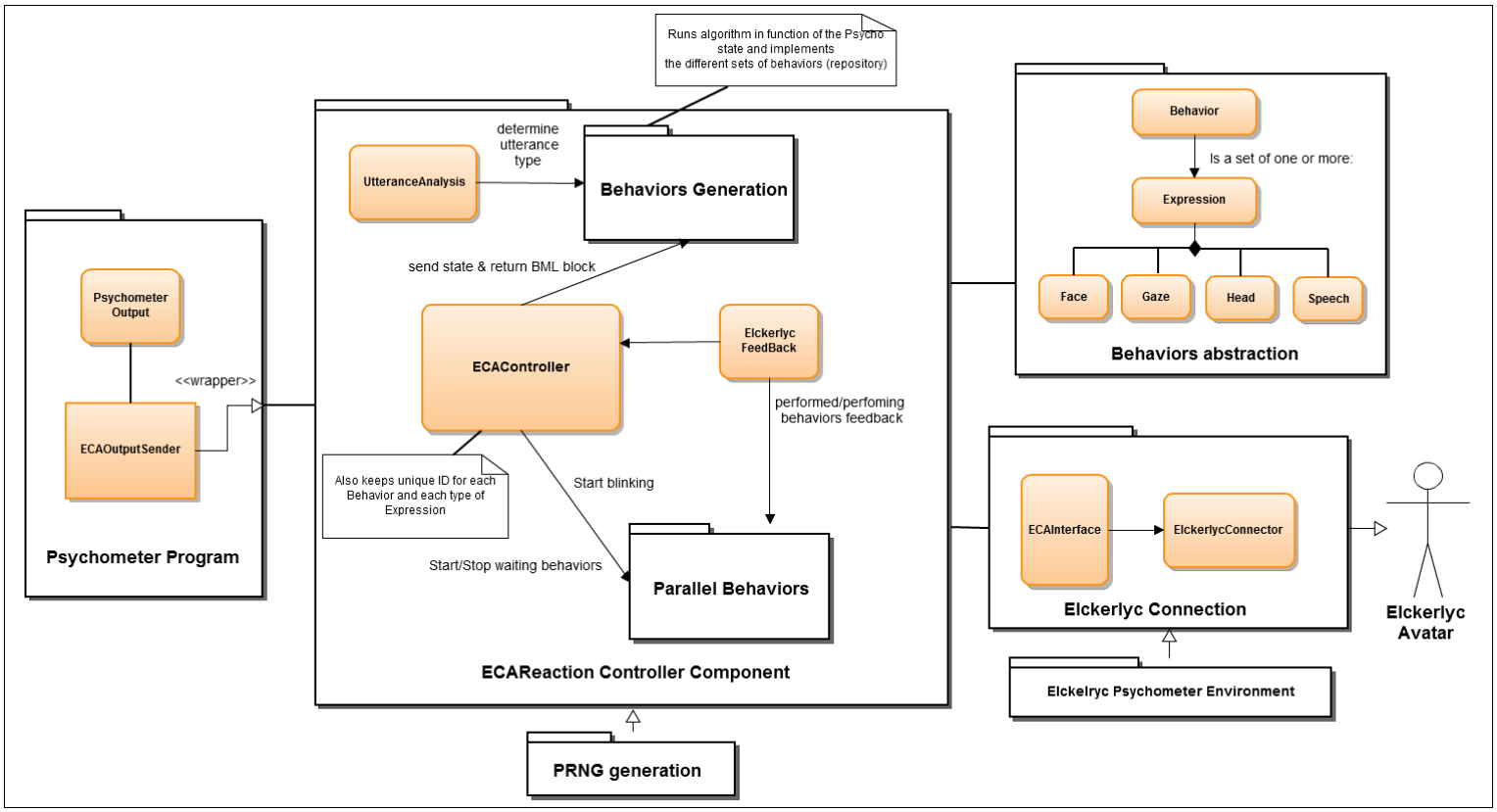

Ilustración 1: Nivel 0 – Diagrama de descomposición

Módulo ECAReaction Controller Component

Controlador principal, dirigido por la clase ECAcontroller.java, que captura el estado e información relativa del Psychometer (si hay un habla, tipo, tiempos, etc.) y también almacena información para futuras ejecuciones. Dependiendo del estado actual llama a la ejecución de un algoritmo u otro dentro de la clase BehaviorStates.java. Este módulo esta dividido en varios paquetes que incluyen el controlador principal, la implementación del feedback de Elckerlyc (información acerca de los comportamientos en procesamiento y que fueron procesador), la ejecución de comportamientos series y comportamientos paralelos (como parpadeo, mirada aleatoria, comportamientos de espera, etc.).

Las clases y/o paquetes principales de este módulo son: BejaviorStates.java, BehaviorCollection.java, UtteranceAnalysis.java y ParallelBehavior package.

Módulo ECAExpressions

Este paquete incluye los comportamientos y expresiones representados en java (independientes del lenguaje BML, generados mediante el comando toString()), métodos de sincronización entre comportamientos y/o entre expresiones. La manera de representar los comportamientos y expresiones en java ha permitido encapsular el objeto pertinente del lenguaje BML.

Las clases y/o paquetes principales de este módulo son: BehaviorSet.java, Identifier.java, ExpressionSync.java; y los paquetes Expressions y Behaviours.

Módulo ECAInterface

Este módulo es el encargador de la comunicación entre Elckerlyc y el Psychometer. Utiliza una interfaz, llamada ElckerlycConnector.java que es necesaria para comunicar el motor de Elckerlyc con la entrada del programa (en el caso del Psychometer). Inicialmente ElckerlycConnector estaba orientado para ser utilizado únicamente mediante una comunicación TCP con Elckerlyc. Finalmente se permite esta comunicación y trabajar directamente con una clase llamada RealizerBridge. El RealizerBridge ofrece acceso al avatar de Elckerlyc mediante una interfaz propia. De forma que este paquete permite trabajar de las formas, importando la interfaz propia de Elckerlyc o bien estableciendo una comunicación TCP.

Las clases principales de este módulo son: IEmbodiedConversationalAgent.java, ElckerlycConnection.java t ECAInterfaceException.java

Módulo PRNGUtils

Este módulo permite una generación de pseudo-números distribuidos de forma relativamente uniforme en un rango y con una diferencia mínima entre cada número aleatorio generado. El sistema emplea muchos valores pseudo -aleatorios y este módulo permite aportar más naturalidad al generar mejores número que las clase Random incluida en la API de Java.

Las clases principales de este módulo son: MTRandom.java y RandomGeneration.java

1.3 Resultados del Proceso de Validación

La validación realizada consistió en dos grupos de estudiantes de la universidad de Twente, el primer grupo, denominado grupo “A”, ejecutó el sistema final; mientras que el grupo B ejecutó un sistema que lanzaba los comportamientos de forma pseudo-aleatoria. Los objetivos de la evaluación era medir cómo los participantes percibían cada uno de las versiones en términos de usabilidad y atractivo (en el sentido de si les parecían “naturales” los comportamientos generados por la ECA). Estos resultados se obtuvieron mediante la evaluación AttrakDiff, explicada brevemente más adelante.

Durante el desarrollo de cada evaluación se trató de extraer impresiones superficiales de la interacción de cada participante con el sistema mediante la grabación de cada participante interaccionando con el sistema, como tiempo dedicado a mirar al avatar o al cuadro de dialogo (salida mostrada por el Psychometer), o el tiempo que dedicaban a responder cada pregunta o a buscar definiciones de posibles palabras que no entendieran. Al finalizar cada sesión con el sistema cada participante rellenaba un la evaluación AttrakDiff que consiste en una serie de palabras pares opuesta con graduación a elegir, por cada par, de uno a siete a seleccionar por el participante. AttrakDiff fue elegido como medio para la evaluación por varios motivos. En primer lugar porque implementa el modelo teórico investigado y validado en varios estudios por Hassenzahl M., quien ha investigado durante décadas nuevas áreas para evaluar la experiencia del usuario y percepciones, en especial en el ámbito de la interacción del ser humano con los computadores. En segundo lugar, AttrakDiff automatiza el proceso para el evaluador, procesando las preguntas y generando varios informes finales y la confidencia de estos resultados. Además, el tiempo requerido por cada participante para rellenar la encuesta es muy corto, menos de 10 minutos.

Después de la evaluación de cada participante se le solicitó que rellenase un formulario con unas preguntas acotadas aparta. Por último, se contrastaba la información obtenida con los videos grabados y se anotaba cualquier información relevante así como analizar los posibles casos extremos.

De los resultados que se obtuvieron se pudo extraer varias conclusiones. A partir de la evaluación de AttrakDiff se puede asegurar que la versión A (resultante de este proyecto) es más atractiva que la versión B, donde los comportamientos son generados de forma aleatoria. Pero, por otra parte, no se puede afirmar dentro del rectángulo de confidencia que la experiencia de usuario y la usabilidad son mejores en la versión A que en la versión B, aunque se observa que, disparar comportamientos que los usuarios pueden esperar, mejoran estos valores sin lugar a dudas.

Como opinión personal la experiencia del usuario puede ser mejorable. Aunque en la versión final se integró la ventana de preguntas con la ventana del avatar, así como se creó una oficina virtual para el avatar; estas modificaciones no fueron incluidas durante el proceso de validación y, en mi opinión, habrían mejorado estos resultados de forma notable.

En cuanto al atractivo del proyecto AttrakDiff lo valoró como “muy atractivo”, que pudo ser contrastada con la información respondida de forma individual por cada participante, donde la versión A obtuvo una valoración de 4.1 sobre 5; mientras que la versión B obtuvo un valor de 2.9 sobre 5. Por lo tanto se puede concluir que los participantes prefirieron la versión A sobre la B.

Respecto al habla de la ECA, un tema que se temía perjudicaría los resultados por la pobre calidad ofrecida en la versión del modulador utilizada, no fue apreciada como tal por los participantes; aunque si notaron que era un área a mejorar.

En resumen, para mejorar notablemente la usabilidad del sistema (recordando la idea de usabilidad como las tres “es” (en inglés): fácil de usar, fácil de aprender, fácil de recordar. La fase dos del proyecto habría mejorado esta área debido a que, precisamente, un agente conversacional es más útil cuando aporta un feedback emocional en cada momento al usuario. Pero dicha implementación tiene que hacerse con sumo cuidado para evitar el resultado opuesto al deseado: un agente demasiado persistente puede molestar al usuario (nótese el famoso ejemplo del agente “clip” de Microsoft Word). Por supuesto, si se puede mejorar a la vez la voz de la ECA, el atractivo y la naturalidad percibida por los usuarios se incrementaría también.

1.3 Conclusiones del Proyecto

El objetivo del Lenguaje de Marcado de Comportamiento (BML) es crear un marco de representación en tiempo real para la generación de un comportamiento multimodal de los agentes conversacionales. Desde el comienzo de la especificación BML en el año 2006 se han propuesto las bases del conocimiento que describen la forma y la generación de un comportamiento multimodal de comunicación en los diferentes niveles de abstracción. Los niveles "representan las interfaces entre las etapas de (1) la planificación de la intención comunicativa, (2) planificación de la realización multimodal de esta intención, y (3) la realización de las conductas previstas". Mediar entre las dos primeras etapas es la Functional Markup Language (FML), que describe la intención sin hacer referencia a la forma de la superficie; la FML en gran medida sigue siendo indefinida. Entre las dos últimas etapas se encuentra el Behavior Markup Language (BML), que describe el comportamiento humano no verbal y verbal de una manera independiente de la realización particular (animación) utilizada. Durante la investigación de este proyecto, basado en la aplicación del marco BML en el proyecto Elckerlyc, se ha demostrado el progreso realizado en la aplicación del BML. Sin embargo, se requiere un impulso es la especificación funcional. En otras palabras, ha habido un gran avance en la especificación de "lo que la ECA debe hacer" mediante el lenguaje BML, pero hay todavía mucho por hacer en la especificación de "cómo debe ser hecho por la ECA". Este hecho se demostró en las áreas de la expresión y generación de voces y algunas de las limitaciones de tiempo durante el diseño de los comportamientos de la mirada.

Centrándonos en este proyecto, algunos comportamientos realizados por la ECA ofrecida por Elckerlyc se han desarrollado como un puente para conectarse con el Psychometer, utilizando como base los diferentes estados dentro del Psychometer y por los que se pasa en diferentes momentos durante su ejecución (recordar que el Psychometer es un programa secuencial basado en una interacción paso a paso con el usuario). El resultado ha sido lo que a lo largo de la investigación se llamó la generación de comportamientos secuenciales. Como se ha demostrado durante el desarrollo de esta investigación, la forma secuencial, no tasa para las conductas multimodal en tiempo real. Por lo tanto, al introducir comportamientos más naturales se han implementado dos hilos para hacer frente al tiempo real de la generación de conductas (lo que se llama las conductas de espera, ejecutado después de un tiempo pseudo-aleatorio de espera para la próxima interacción con el usuario y el comportamiento intermitente, que es independiente de la situación actual interna y la interacción del usuario durante la ejecución). El principal objetivo era simular el comportamiento humano en la ECA, por ejemplo, la tendencia al apartar la mirada de los oyentes o potenciar dicha mirada cuando quieren seguir hablando y dar una información al usuario. Para hacer posible esta convivencia entre la generación secuencial de comportamientos y la generación paralela, el algoritmo propuesto ha sido modificado para adaptarla a las circunstancias actuales, pero en esencia sigue la misma filosofía comentada en el apartado "algoritmo" de la tesina. Este algoritmo muestra ser un buen método para aplicaciones de paso a paso. Sin embargo, hay que hacer una importante mención: este algoritmo se basa en la suposición de que existe un control total sobre el tiempo, pero la realidad es que Elckerlyc, como la especificación BML dice, hace lo mejor posible para satisfacer las limitaciones de tiempo. En consecuencia, como desarrollador, no es posible

lograr tiempos exactos (por ejemplo, dos mensajes con exactamente la misma sintaxis BML puede diferir en el tiempo cuando son realizadas por el avatar).

Ha habido dificultades en este proyecto de investigación, como consecuencia directa de la utilización del proyecto de desarrollo Elckerlyc, que se explican en la sección 5 de la tesina. La principal consecuencia es que este proyecto sólo terminó con la primera etapa de las fases iniciales propuestas. La segunda fase habría incrementado la naturalidad de las conductas realizadas y daría al usuario una mejor retroalimentación con la información de expresiones actuales (por lo que la experiencia del usuario sería más rico). Una herramienta de depuración del sistema Elckerlyc es necesaria y no tenerla puede inducir a errores que no se esperaban y, más grave, se carece de una forma de arreglarlos. La buena noticia es que todo el esfuerzo invertido ayudará al equipo Elckerlyc para mejorar el sistema y los futuros usuarios, para tener un mejor conocimiento del sistema. Y los beneficios personales, como haber tenido la oportunidad de tratar con un proyecto tan grande como Elckerlyc.

Es importante mencionar que me vi obligado a rehacer diferentes partes de Elckerlyc que no funcionaban o estaban pendientes de su implementación (como por ejemplo el mundo Elckerlyc específico, se explica en la sección 4) o la adaptación de las restricciones impuestas por Elckerlyc en algunas circunstancias. En muchas ocasiones, hay una falta entre el qué y cómo es posible que desee obtener algunos de los comportamientos de Elckerlyc, y lo que realmente es capaz de hacer. Lograr un buen comportamiento en el tiempo no es tan simple, y definir los criterios de "bueno" también. No era sólo fue una cuestión de investigar el diseño de los comportamientos que deseaba, pero también había que contemplar las posibilidades de Elckerlyc en cada nueva versión, y adaptar estos comportamientos al entorno de una manera coherente.

A pesar de estas limitaciones, como podemos ver en la evaluación los resultados son bastante buenos. También se debe indicar decir que hay otros aspectos que los tiempos de los comportamientos que tienen más influencia en los comportamientos resultantes y que, en general, las conductas percibidas por los participantes no son tan malas como se esperaba inicialmente. Sin embargo, hay un considerable espacio para mejorar.

En mi opinión persona y a partir de la información aportada por las participaciones recomendaría algunas mejoras para aumentar la usabilidad del sistema y la naturalidad de la ECA. En primer lugar, en vez de usar dos ventanas para el sistema (una de las ventanas de diálogo y otro para la ECA) debe haber una fusión entre estas dos ventanas. Por ejemplo, la ECA se podría situar en el área principal de la ventana y, a efectos informativos, la ventana de diálogo se podría situar por debajo de la ECA. Sin embargo, para que esto sea posible un nuevo gestor de ventanas debería ser implementada y recodificada a la nueva versión de Java. Otras áreas de mejora son, por supuesto, el rediseño de los comportamientos actuales y la investigación de la fase dos, que estaba proyectado como parte de este proyecto. Esto daría una información adecuada al usuario en todo momento, mejorando la experiencia del usuario.

Hubo algunas ideas notables dadas por los participantes que recomendarían mejorar otras áreas. Por ejemplo, el diálogo debe tener un retraso sincronizado con la ECA (o tal vez las palabras pueden aparecer al mismo tiempo que son procesados la ECA). Esto no era posible antes, pero ahora con la nueva implementación FML se puede especificar la voz y controlarla

para cada palabra y fonema emitido por la ECA. Otra mejora interesante sería el diseño de un espacio virtual donde la ECA cobre "vida". Por ejemplo, una habitación psicológica (como la generada más tarde y mostrada en el vídeo) debería ser incluida en la validación para mejorar la inmersión del usuario en el programa. Por último, sería ideal poner en práctica el reconocimiento de voz para complementar la entrada, en vez del teclado. Obviamente, esto no es una tarea fácil y los mejores resultados se obtienen en los sistemas que requieren un aprendizaje previo para cada usuario que desea mantener un diálogo con la ECA.

¿Y qué pasa con el sistema Psychometer en su última versión? La evaluación ha mostrado algunas mejoras que se deben aplicar a este sistema. Más palabras deben ser incluidas en el conjunto actual que es reconocido a la hora de responder las preguntas y la inclusión de la entrada numérica debe ser permitida durante toda la ejecución (no sólo cuando el usuario introduce un texto erróneo). Por último, la repetición de preguntas similares, debería ser controlada. Es cierto que cuando las preguntas se repiten la frase está formulada de una manera diferente, pero algunos de los participantes se han quejado de este hecho. Supongo que este tipo de repeticiones es importante para asegurar la validez de las respuestas. Pero en mi opinión, estas repeticiones se intercalan con otras preguntas. Y con esto finaliza las sugerencias para mejorar el sistema.